Criar um aplicativo de estoque com React Native usando o Expo chamado appartlight contendo os recursos adição/remoção de produtos, visao de estoque e uma base de dados local.

**O ChatGPT disse:**

ChatGPT

Para criar o aplicativo de estoque chamado **Appartlight** com **React Native** e **Expo**, vou detalhar o processo com os recursos de **adição/remoção de produtos**, **visão de estoque** e uma **base de dados local** usando **AsyncStorage**. Vamos seguir os passos descritos anteriormente, mas adaptando para seu projeto.

**1. Configuração do projeto com Expo**

1. Instale o Expo CLI, caso ainda não tenha:

bash

Copiar código

npm install -g expo-cli

1. Crie um novo projeto:

bash

Copiar código

expo init appartlight

1. Navegue até a pasta do projeto:

bash

Copiar código

cd appartlight

1. Execute o projeto:

bash

Copiar código

expo start

**2. Instalação de bibliotecas necessárias**

* **React Navigation** para navegação entre telas:

bash

Copiar código

npm install @react-navigation/native

npm install @react-navigation/stack

npm install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view

* **AsyncStorage** para armazenar dados localmente:

bash

Copiar código

expo install @react-native-async-storage/async-storage

**3. Estruturação das telas**

As telas principais do Appartlight serão:

* **HomeScreen.js**: Tela principal com a visão do estoque.
* **AddProductScreen.js**: Tela para adicionar novos produtos.
* **ProductDetailScreen.js**: Tela de detalhes do produto, com a opção de removê-lo.

**Estrutura básica da navegação:**

js

Copiar código

// App.js

import React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

import HomeScreen from './screens/HomeScreen';

import AddProductScreen from './screens/AddProductScreen';

import ProductDetailScreen from './screens/ProductDetailScreen';

const Stack = createStackNavigator();

export default function App() {

return (

<NavigationContainer>

<Stack.Navigator initialRouteName="Home">

<Stack.Screen name="Home" component={HomeScreen} />

<Stack.Screen name="AddProduct" component={AddProductScreen} />

<Stack.Screen name="ProductDetail" component={ProductDetailScreen} />

</Stack.Navigator>

</NavigationContainer>

);

}

**4. Tela de visão de estoque (HomeScreen.js)**

Aqui será exibida a lista de produtos do estoque. Cada item terá um nome e quantidade, e o usuário pode tocar para ver detalhes ou adicionar um novo produto.

js

Copiar código

// screens/HomeScreen.js

import React, { useState, useEffect } from 'react';

import { View, Text, Button, FlatList, TouchableOpacity } from 'react-native';

import AsyncStorage from '@react-native-async-storage/async-storage';

export default function HomeScreen({ navigation }) {

const [products, setProducts] = useState([]);

useEffect(() => {

const unsubscribe = navigation.addListener('focus', () => {

loadProducts();

});

return unsubscribe;

}, [navigation]);

const loadProducts = async () => {

try {

const storedProducts = await AsyncStorage.getItem('products');

if (storedProducts !== null) {

setProducts(JSON.parse(storedProducts));

}

} catch (error) {

console.log(error);

}

};

const renderProduct = ({ item }) => (

<TouchableOpacity onPress={() => navigation.navigate('ProductDetail', { product: item })}>

<Text>{item.name} - {item.quantity}</Text>

</TouchableOpacity>

);

return (

<View>

<FlatList

data={products}

renderItem={renderProduct}

keyExtractor={(item) => item.id}

/>

<Button title="Add Product" onPress={() => navigation.navigate('AddProduct')} />

</View>

);

}

**5. Tela de adição de produtos (AddProductScreen.js)**

Permite adicionar novos produtos ao estoque e salvar no **AsyncStorage**.

js

Copiar código

// screens/AddProductScreen.js

import React, { useState } from 'react';

import { View, Text, TextInput, Button } from 'react-native';

import AsyncStorage from '@react-native-async-storage/async-storage';

export default function AddProductScreen({ navigation }) {

const [name, setName] = useState('');

const [quantity, setQuantity] = useState('');

const addProduct = async () => {

try {

const storedProducts = await AsyncStorage.getItem('products');

const products = storedProducts ? JSON.parse(storedProducts) : [];

const newProduct = {

id: Date.now().toString(),

name,

quantity: parseInt(quantity, 10)

};

products.push(newProduct);

await AsyncStorage.setItem('products', JSON.stringify(products));

navigation.goBack();

} catch (error) {

console.log(error);

}

};

return (

<View>

<Text>Product Name</Text>

<TextInput value={name} onChangeText={setName} />

<Text>Quantity</Text>

<TextInput value={quantity} onChangeText={setQuantity} keyboardType="numeric" />

<Button title="Add Product" onPress={addProduct} />

</View>

);

}

**6. Tela de detalhes e remoção de produto (ProductDetailScreen.js)**

Permite visualizar os detalhes de um produto e removê-lo do estoque.

js

Copiar código

// screens/ProductDetailScreen.js

import React from 'react';

import { View, Text, Button, Alert } from 'react-native';

import AsyncStorage from '@react-native-async-storage/async-storage';

export default function ProductDetailScreen({ route, navigation }) {

const { product } = route.params;

const removeProduct = async () => {

try {

const storedProducts = await AsyncStorage.getItem('products');

let products = storedProducts ? JSON.parse(storedProducts) : [];

products = products.filter(p => p.id !== product.id);

await AsyncStorage.setItem('products', JSON.stringify(products));

navigation.goBack();

} catch (error) {

console.log(error);

}

};

const confirmRemove = () => {

Alert.alert(

"Remove Product",

"Are you sure you want to remove this product?",

[

{ text: "Cancel", style: "cancel" },

{ text: "OK", onPress: removeProduct }

]

);

};

return (

<View>

<Text>Product: {product.name}</Text>

<Text>Quantity: {product.quantity}</Text>

<Button title="Remove Product" onPress={confirmRemove} />

</View>

);

}

Aqui estão as instruções passo a passo para rodar o aplicativo **Appartlight**:

**1. Pré-requisitos**

Antes de começar, certifique-se de ter as ferramentas necessárias instaladas:

* **Node.js** (que inclui o npm). Se você ainda não tem, pode baixar aqui: Node.js.
* **Expo CLI**. Se ainda não tem o Expo instalado globalmente, você pode instalá-lo com o comando:

bash

Copiar código

npm install -g expo-cli

* **Emulador** ou **dispositivo físico**:
  + Um emulador Android ou iOS (como o Android Studio ou Xcode).
  + Ou o app Expo Go no seu dispositivo físico (Android/iOS), que pode ser baixado diretamente das lojas de apps.

**2. Baixar ou clonar o projeto**

Se o código estiver no GitHub ou em algum repositório, faça o clone usando o comando:

bash

Copiar código

git clone <URL\_DO\_REPOSITORIO>

Caso esteja desenvolvendo localmente, navegue até a pasta onde o projeto foi salvo.

**3. Instalar as dependências do projeto**

Dentro do diretório do projeto, rode o seguinte comando para instalar todas as bibliotecas e dependências necessárias:

bash

Copiar código

npm install

**4. Executar o aplicativo**

Após as dependências serem instaladas, inicie o servidor Expo com o seguinte comando:

bash

Copiar código

expo start

Isso abrirá uma janela no navegador com um **QR code** que você pode escanear usando o aplicativo **Expo Go** no seu dispositivo físico.

**5. Rodando no emulador ou dispositivo físico**

* **Dispositivo físico**:
  1. No seu smartphone, abra o app **Expo Go**.
  2. Escaneie o QR code exibido no navegador após rodar expo start.
  3. O aplicativo será executado diretamente no seu dispositivo.
* **Emulador**:
  1. Para Android: Abra o Android Studio, inicie o emulador e no terminal, execute:

bash

Copiar código

expo run:android